

AD-A267 948



1992

THESIS/DOSSIER/ATX/DONK

(Handwritten signature)

A Fuzzy Method for Preliminary Design Random Access
Memory Failure Rate Prediction.

Capt Keith R. Weyenberg

AFIT Student Attending: University of Missouri-Rolla

AFIT/CI/CIA- 93-014

AFIT/CI
Wright-Patterson AFB OH 45433-6583

Approved for Public Release IAW 190-1
Distribution Unlimited
MICHAEL M. BRICKER, SMSgt, USAF
Chief Administration

S **DTIC**
ELECTE
AUG 17 1993
A **D**

20 8 10 1/2 8

93-19059



A FUZZY METHOD FOR PRELIMINARY DESIGN RANDOM ACCESS MEMORY
FAILURE RATE PREDICTION

BY

KEITH ROBERT WEYENBERG, 1960-

A THESIS

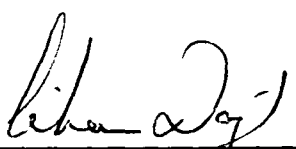
Presented to the Faculty of the Graduate School of the
UNIVERSITY OF MISSOURI - ROLLA
in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

ENGINEERING MANAGEMENT

1992


Cihan H. Dagli (Advisor)


Henry E. Metzger

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Code	
Dist	Avail and/or Special
A-1	


Kelvin Erickson

DTIC QUALITY INSPECTED 3

ABSTRACT

Preliminary design failure rate predictions tend to be rough approximations due to a lack of sufficient information about components, materials, requirements, etc.. Often, experienced engineers can give a qualitative assessment of the possible failure rate of a given design, but have a difficult time quantifying that assessment without the aid of historical data, mathematical equations, or computers. It would be much easier if the experienced engineers judgement or verbal, qualitative, evaluations could be converted to quantitative estimates?

Fuzzy-set theory provides a method to convert qualitative or subjective evaluations into quantitative ones.

This work looks at how fuzzy-set theory can be used to convert qualitative evaluations of factors affecting failure rates into quantitative estimates of failure rates. By using fuzzy estimates of the environment and design effects on failure rate, data is generated by the fuzzy failure rate prediction method.

The data generated by this method compares favorably with the data source, MIL-HDBK-217E, Reliability Prediction of Electronic Equipment.

ACKNOWLEDGEMENTS

I would like to express sincere appreciation to my advisor Dr. Cihan Dagli and the other members of my committee, Dr. Henry Metzner and Dr. Kelvin Erickson for their support in completing this project. I would also like to thank Dr. Dagli for his patience during my futile search for actual field reliability data on missile systems in sufficient detail to accomplish this study. I would also like to thank Dr. Metzner for his guidance on analysis methods and data needed for a reliability study.

I especially want to thank the Air Force for giving me the opportunity to attend graduate school at the University of Missouri-Rolla (UMR).

Most of all I would like to thank my wife, Sheila, daughter, Valerie and son, Michael who have endured my many long days and nights of studying. It is their love and support, that has allowed me to complete my degree and enjoy my studies at UMR.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
LIST OF ILLUSTRATIONS	vii
LIST OF TABLES	viii
SECTION	
I. INTRODUCTION	1
A. RELIABILITY PREDICTION OVERVIEW	1
B. FUZZY LOGIC INTRODUCTION	2
II. LITERATURE REVIEW	5
A. RELIABILITY PREDICTION LIMITATIONS	5
B. APPLICATIONS OF NEURAL NETWORKS AND FUZZY SYSTEMS	6
1. Backpropagation Neural Network	7
2. Fuzzy Associative Memory (FAM)	9
III. INITIAL INVESTIGATION WITH BACKPROPAGATION NEURAL NETWORK	13
A. INTRODUCTION	13
B. DATA DEFINITION	14
C. NETWORK DEFINITION	15
D. NETWORK TRAINING	16
E. RESULTS	17
F. OBSERVATIONS	21
IV. THE FUZZY MODEL	24
A. FUZZY KNOWLEDGE REPRESENTATIONS	24
B. SHAPES OF THE FUZZY-SETS	26

C. BIOFAMs CORRELATION-MINIMUM INFERENCE. . .	30
D. BIOFAM DEFUZZIFICATION	36
E. MODEL SUMMARY	38
V. TESTING, RESULTS AND FUTURE WORK	40
A. EARLY MODEL DESIGN RESULTS	40
B. CURRENT MODEL RESULTS	42
C. FOLLOW-ON STUDY SUGGESTIONS	44
VI. CONCLUSION	46
APPENDICES	
A. BACKPROPAGATION DATA: TABULAR LISTINGS	48
B. MIL-HDBK-217E DATA	55
C. HISTOGRAM OF TABLE XIII DATA	58
D. COMPARISON OF MODEL DATA TO MIL-HDBK-217E DATA	60
E. PASCAL PROGRAM FOR FUZZY FAILURE RATE PREDICTION	62
F. DATA FILES USED BY THE MODEL	73
BIBLIOGRAPHY	75
VITA	77

LIST OF ILLUSTRATIONS

Figure		Page
1	Backpropagation Neural Network Architecture. . . .	8
2	Fuzzy-Sets for Environment.	28
3	Fuzzy-Sets for Design.	29
4	Failure Rate Fuzzy-Sets Part 1.	32
5	Failure Rate Fuzzy-Sets Part 2.	32
6	Failure Rate Fuzzy Sets Part 3.	33
7	Correlation-Product Encoding of FAM Rule 8. . . .	33
8	Correlation-Product Encoding of FAM Rule 13. . . .	34
9	Fuzzy Centroid Determination (Defuzzification). .	35
10	Histogram of Table XIII Data.	59

LIST OF TABLES

Table	Page
I Training tolerance levels achieved.	17
II Comparison of network output to expected output for the network trained to an error tolerance of 0.1.	19
III Comparison of expected output to network output for data sets giving the log of the expected output.	20
IV Linguistic variables for environment effects.	25
V Linguistic variables for design.	25
VI Linguistic variables for failure rate.	26
VII Matrix of FAM rules.	27
VIII 70 item raw data set.	49
IX 16 item raw data set.	52
X 14 item data set with 100,000 and higher MTBFs removed.	53
XI 16 item data set with zeros changed to 0.2 and output is log of desired number.	54
XII MIL-HDBK-217E environment abbreviation definitions.	56
XIII Generic failure rates for MOS dynamic RAMs.	57
XIV Comparison of model failure rate predictions to Table XIII.	61

I. INTRODUCTION

A. RELIABILITY PREDICTION OVERVIEW

A typical failure rate prediction for an electronic component would involve going to an electronic component failure rate reference such as MIL-HDBK-217E, Reliability Prediction of Electronic Equipment, finding factor values for things like environment and quality, finding a base failure rate, determining how many elements make up the component, then putting the appropriate numbers into an equation developed for that particular component type. This would involve finding 3 or 4 tables from hundreds.

The example above points out one of the limitations of reliability predictions, the mechanics of the process. The part stress analysis method, the primary method of prediction used in MIL-HDBK-217E, requires a great amount of detail. This detail imposes a time and cost penalty. More importantly, these details are usually not available in the early design stages. Therefore, a simpler method is also covered, the parts count method[1].

Reliability predictions of components and systems are an essential step in the design process. These predictions provide the rationale for decisions on alternate design paths, part quality choices, levels of derating, level of redundancy, choosing between using proven technology and equipment versus new or state-of-the-art technology and equipment, and other

design and development related issues[2]. But what do these predictions really tell us?

Reliability has been defined as the probability that a product performs its intended function for a stated period of time under specified operating conditions. This means that it is a quantitative assessment based on the measure of two quantifiable characteristics, time and operating conditions[3].

A great deal of work has gone into defining the effects of design variations and operating conditions on reliability. Much of this work is published in a number of military standards, military handbooks, and other government documents. These works provide engineers with the history needed to make predictions about the failure rates of various products and designs with little if any need for engineering judgement. In many engineering applications, however, it is difficult to evaluate the probabilities and consequences from past experiences, because of the dynamic environments of systems, and especially because there are situations where past experience does not exist[4]. In these cases, judgement and expertise can play an important role in reliability prediction.

B. FUZZY LOGIC INTRODUCTION

Judgement inherently contains a level of uncertainty. Engineering analysis is also subject to uncertainties[5]. In reliability analysis these uncertainties exist in the

estimates of operating conditions, variations in historical failure rates, cycle times, and quality of the design. Experts can qualitatively define uncertainties, such as higher or lower, but have a difficult time quantifying these terms.

The key elements of human thinking are not numbers but labels of fuzzy-sets[6]. Humans are more efficient in qualitative evaluation than in quantitative analysis. The knowledge of an experienced reliability engineer usually consists of qualitative variables stated verbally. If forced to give numerical estimates, humans tend to get biased because they are taxed to operate in a mode which requires more mental effort and guessing. Fuzzy-set theory makes it possible to quantify and manipulate qualitative statements, vagueness, or subjectivity of opinion[7]. It would be much easier to develop preliminary design failure rate predictions using expert judgement rather than going through all the tables and calculations in the failure rate handbooks?

Fuzzy-set theory provides the method to convert qualitative judgements to quantitative estimates of failure rates. By combining the judgement of the engineer and fuzzy-set theory, we can account for the uncertainties of reliability prediction. Then, through the use of a computer program, predictions can be generated easier and enhance the application of the reliability prediction to varying conditions.

Therefore, the fuzzy approach to the example in the opening paragraph would be to ask the engineer for a

linguistic evaluation of the environment and quality, apply that evaluation to the fuzzy-sets associated with environment and quality, and determine a factor to apply to the base failure rate. This method provides the basis for this study.

An industry standard on electronic failure rate prediction, MIL-HDBK-217E, contains data in the format just described. The model developed here predicts failure rates for metal oxide semiconductor, dynamic random access memory (MOS, DRAM) chips with values that are quite close to those developed in MIL-HDBK-217E.

In the next section, similar work conducted in related areas will be reviewed. First, some of the reliability predictions limitations will be discussed. Then, the development of fuzzy-sets in similar applications will be examined. In Section III, the results of an earlier attempt to model failure rates with a backpropagation neural network will be reviewed. Section IV will describe the fuzzy associative memory (FAM) model. Finally, the results and observations of the fuzzy approach to failure rate prediction will be presented along with suggestions for further research in this area.

II. LITERATURE REVIEW

A. RELIABILITY PREDICTION LIMITATIONS

Reliability predictions provide a qualitative assessment of how well a given design will perform its intended function under some specified conditions. This assessment can be used to evaluate proposed design changes, define special environmental control limits, identify special maintenance or handling procedures, evaluate the significance of reported failures, or to perform other engineering analyses. Like any engineering tool, reliability predictions must be used intelligently, with consideration given to their limitations[1].

One limitation is that failure rate models are point estimates, based on the best data available at the time of the predictions. Therefore, they are only valid for the conditions under which they were developed, and only for the product analyzed. Some extrapolation is possible, but the empirical nature of the models severely limits the degree to which data can be extrapolated[1].

Another limitation is the dynamic nature of electronic technology. The rapid growth of new electronic devices and processes make reliability predictions increasingly more difficult. This requires continual change or improvement in prediction methods and procedures. As an example, a revolutionary design may defy current prediction methods requiring a revolutionary prediction technique.

The final limitation is reliance of reliability predictions on correct application by the user. A correctly applied reliability prediction and reliability model will be a very useful tool for an engineer. If the user of the reliability prediction is only concerned about a reliability number, he will usually find a way to achieve it, without any impact to the system in question[1]. This is of little benefit to the engineer, the company, or the product.

Given these limitations, we have to ask if there is mathematical method other than classical statistics that could provide reliability predictions quickly and easily when these limiting conditions change. Two areas worth studying are neural networks and fuzzy-set theory.

B. APPLICATIONS OF NEURAL NETWORKS AND FUZZY SYSTEMS

Neural networks and fuzzy systems estimate functions from sample data as do statistical and artificial intelligence (AI) approaches. For each problem, statistical approaches require one to guess how outputs functionally depend on inputs. Neural and fuzzy systems do not require a mathematical model, as such, and are model-free estimators[8].

Neural networks can be embedded in the mathematical fields of dynamical systems, adaptive control, and statistics. Adaptive neural algorithms have been used in high-speed modems, long-distance telephone calls, and some airport bomb detectors. Fuzzy theory overlaps with the neural network fields and with probability, mathematical logic, and measure

theory[8]. Since the introduction of fuzzy-set theory by Lotfi Zadeh in 1965[9], fuzzy systems have been adopted for use in running subways, tuning televisions and computer disc heads, focusing and stabilizing camcorders, adjusting air conditioners and washing machines, defrosting refrigerators, scheduling elevators and traffic lights, and controlling automobile motors, suspensions, and emergency braking systems.

1. Backpropagation Neural Network. Two of the primary applications of neural networks are situations where only a few decisions are required from a massive amount of data and situations where a complex nonlinear mapping must be learned[10]. This study was looking for a complex nonlinear mapping. The investigation into a neural network failure rate prediction was conducted using a backpropagation neural network. The backpropagation neural network consists of a number of parallel distributed processing elements, all interconnected as shown in Figure 1. This network consists of 3 or more layers of processing elements. The backpropagation neural network is a heteroassociative, function-estimating artificial neural system that stores arbitrary analog spatial pattern pairs (A_k, C_k) , $k = 1, 2, \dots, m$, using a multilayer gradient descent error-correction encoding algorithm, where the k th pattern pair is represented by vectors $A_k = (a_1^k, \dots, a_n^k)$ and $C_k = (c_1^k, \dots, c_q^k)$ [10]. Where the A_k vector is the input to the network, and the C_k vector is the output. In other

words, a backpropagation network which is properly trained, can read an input A_k or close to A_k and provide an output C_k .

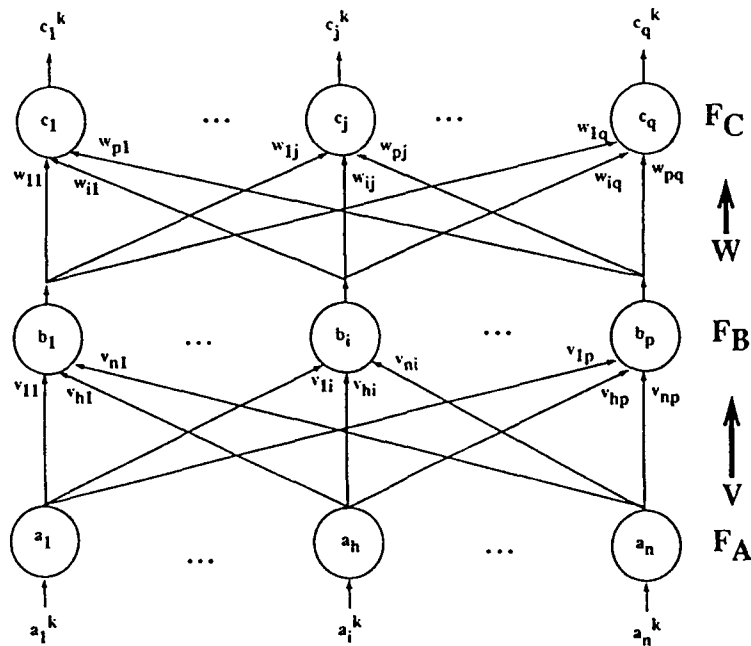


Figure 1 Backpropagation Neural Network Architecture.

The initial investigation tried to correlate the number of active elements to the mean-time-between-failure (MTBF) in electronic equipment. The thought behind this approach is that the failures are not random, but nonlinear, and therefore, a backpropagation network could learn or estimate a function that would match that of the failures. The data analyzed came from a Reliability Analysis Center document, Electronic Equipment Reliability Data[11]. This approach was a limited attempt to develop a failure rate model for electronic systems

that resulted in limited success. The results of this study is presented in Section III.

2. Fuzzy Associative Memory (FAM). Mathematical models are never totally exact and accurate. This is because one does not know the true values of the numerical parameters for the problems. One reason one does not know the values for these reliability models is material, processes and environmental conditions change so rapidly that not enough field data can be collected for a thorough study of the failures. One of the main difficulties with failure rate models is once one knows the detailed physics well enough to model it, one generally knows enough about the problem to cure it. Therefore, the reliability problems dealt with in practice tend to be ones that are not well modeled[12].

Fuzziness can be viewed as an alternative to randomness for describing uncertainty. An element belongs to a multivalued or "fuzzy" set to some degree in the range of real numbers from 0 to 1. An element belongs to a nonfuzzy set one hundred percent or zero percent, 1 or 0. Fuzziness measures the degree to which an event occurs, not whether it occurs. Randomness describes the uncertainty of the fact an event will occur or an event will not occur. Whether an event occurs is random and the degree to which an event occurs is fuzzy[8].

One routinely uses fuzzy measures in everyday existence: fast computers, small errors, big trees, partly sunny, almost home, weak batteries. One understands that these terms or

phrases have gray areas where the opposite or converse is also true. Engineers use fuzzy measures also when verbally evaluating designs: very good, weak, robust, bad. If these evaluations can be converted into numerical values, we quantify engineering expertise and judgement. Fortunately, fuzzy-set theory provides the method for converting these linguistic fuzzy values into deterministic ones. The ability to go from a fuzzy measure to a deterministic value has spawned a number of papers on the application of fuzzy-set theory.

There are only a couple papers available that address the use of fuzzy sets with reliability predictions. In 1988, Kubic and Stein addressed chemical process reliability and unreliability using fuzzy sets to determine several factors used to control benzene-toluene distillation[5]. They concluded that fuzzy reliability and unreliability can mimic the decision process of an engineer and can be used to distinguish between feasible designs, those with high reliability and low reliability; infeasible designs, those for which the opposite is true; and promising designs, designs which are economically desirable but, reliability and unreliability values are nearly equal.

An earlier paper by Naruhito Shiraishi and Hitoshi Furuta, looked at the use of fuzziness in determining structural reliability[13]. Their method used fuzzy mathematics to determine fuzzy probability density functions for parameters exhibiting uncertainty. One of their

conclusions was fuzzy probability and subjective assessment can be merged with objective assessment in the calculation of failure probability. They also concluded that fuzzy probability can provide a feasible base to determine the characteristics of correction factors.

Using Kubic and Steins idea that fuzzy sets can be used to determine reliability factors, this study looks at using fuzzy sets for environmental and design characteristics. Verbal inputs describing the possibility of failure are used for each characteristic. These inputs activate the collection of fuzzy sets or rules to different degrees. This information is used to determine the output fuzzy rule through an IF THEN process. The output fuzzy rule is defuzzified and the result is a failure rate prediction.

Using Table 5.2-9 from MIL-HDBK-217E, fuzzy rules were designed to see how well a fuzzy failure rate prediction method would match the failure rates given in the table. The fuzzy failure rate prediction compared quite well to the values found in the table. The details of this comparison are presented in Section IV.

The next section reviews the investigation into failure rate prediction with a backpropagation neural network. It starts with a brief discussion of the backpropagation software. Then, a definition of the data used is presented. After the data is defined, the network design is described. Following the network design, the network training methods and results are reviewed. The results of the training and testing follow the network training section. Finally, some observations about this approach and the results are presented.

III. INITIAL INVESTIGATION WITH BACKPROPAGATION NEURAL NETWORK

A. INTRODUCTION

This study was conducted to see if there was any correlation between the number of active elements and the MTBF for electronic equipment. Data from the report, Electronic Equipment Reliability Data, Summer 1986, suggested a relationship between failure rates and the number of active elements does exist[11]. A model was set up using a backpropagation neural network. The model was designed to predict the MTBF for electronic equipment based on the data presented in the report. A commercial backpropagation computer program, Brain Maker, from California Scientific Software was used for the model[14].

Brain Maker is essentially a flexible neural network shell that allows the user to define a backpropagation neural network to meet his requirements. The user can define the number of input and output processing elements, the type of transfer function used by the processing elements, the learning rate, a noise level to help avoid local minimums, the error tolerance for training and for running the network, and customizable displays to monitor the progress of network training and compare the network output to the expected output. The architecture of this network, as defined by Simpson, is shown in Figure 1 on page 8 [10].

B. DATA DEFINITION

The data from the report mentioned in the previous section analyzed 406 items. The equipment was categorized by type, and the environment in which it was employed. Further information is given on the size of each item, the number of active elements, the number of relevant failures, the number of nonrelevant failures, and the number of hours of operation over which the failures occurred.

The analysis started with 70 different items. The active elements are categorized by 5 different classes of elements, total number of integrated circuits, total number of tubes, total number of discrete, and the total number of active elements for a total of 9 different inputs for the backpropagation network.

The output data was fielded operational MTBF values. This was done to ensure that mature designs with proven reliability were being used. The output values ranged from 64.5 to 504,697 hours MTBF. The total number of active elements ranged from 0 to 40,552. The size of the items and the environments they were used in were mixed. Tabular listings of the data for training the network are presented in Appendix A.

The data was also reduced to a 16 item subset of the original 70 items mentioned above. This subset was composed of items that were all used in ground based systems. The range on MTBF and active elements was the same as the 70 item set. This subset was created to see if environment had any effect on the data and the networks ability to train.

A third division of the data resulted when the items with an MTBF of greater than 100,000 hours were removed from the 16 item data set. This was done to see if the accuracy of the network training could be improved. It is possible the range of MTBF values may have been making it difficult or impossible for the network training to converge with the current network design. Appendix A also contains this data.

The next iteration of data sets occurred with the 14 item data set. The log of each input value was taken along with the log of the expected output. Again, this was done to try and improve the error tolerance during training.

Iteration five changed all the zero's to 0.2 and iteration six took the log of the inputs and expected outputs in the 70 and 16 item data sets. An improved training tolerance was again the goal. Iteration six produced the best results.

C. NETWORK DEFINITION

The network used 9 input processing elements, the hidden layer (layer 2) size ranged from 3 to 10 processing elements as shown in table 1, and one output processing element was used. The inputs were originally scaled from 0 to 1, but the network took an extremely long time to train, if it would train at all. Therefore, all zero inputs were changed to 0.2 which did improve the network performance. Using the 16 item data set and varying the hidden layer size worked well with 5, 6, and 7 hidden layer processing elements. Other hidden layer

sizes did not train as well, or to a training tolerance as low as the 5, 6, and 7 element hidden layers. As a result, a majority of the training runs were conducted around the 5, 6, and 7 hidden layer sizes.

D. NETWORK TRAINING

Most of the effort on this part of the study concentrated on training data sets. Scaling MTBF data that ranges from approximately 65 to over 500,000 to a range of 0 to 1 requires that the network be accurate to about 0.01 before meaningful data can be extracted from it. This level of accuracy could not be obtained with the data and network design used. The lowest training tolerance achieved while still getting a trained network in less than 20 hours (on a 33 MHz 386 PC) was 0.06. This training level was achieved with the 14 item data sets that used the log of all inputs and the log of the expected outputs, and the 16 item set that used the log of the expected outputs only.

The raw data sets with zeros changed to 0.2, and scaled between 0 and 1, were able to train to a level of 0.09. Attempts to train these data sets to levels of 0.08 were unsuccessful when allowed to run for 20 hours. The training variations tried on this data are shown in Table I.

Table I Training tolerance levels achieved.

Hidden Layer Size	3	4	5	6	7	8	9	10
S.R.D. 14 Item			0.09	0.09				
S.R.D. 16 Item	0.2	0.09	0.09	0.09	0.09	0.10	0.09	0.09
S.R.D. 70 Item	X	X	X	X	X	X	X	
S.L.D. 14 Item			0.06				0.06	
S.L.D. 16 Item		0.07	0.06	0.06	0.06		0.06	
S.L.D. 70 Item		0.50	0.40	0.40	0.40	0.40	0.40	0.40

S.R.D. = Scaled raw data with zeros changed to 0.2.

S.L.D. = Scaled, log of expected output data with zeros changed to 0.02.

X = Network did not train within 20 hours.

The numbers in the cells are the training tolerance levels achieved.

E. RESULTS

Due to resource limitations, the training limit was set at 20 hours. The 16 item data set was run for about 48 hours, but the network did not train. This data set contained zeros in the input data. As stated in the Network Training section, the data that had all the zeros changed to 0.2 trained to a

tolerance level of 0.09 using 5 hidden neurons. This was the lowest tolerance level the network would train to within a 20 hour training limit.

A comparison of the expected outputs and the network outputs (in MTBF hours) obtained by presenting the 14 item training set to the network trained with the 16 item data set is shown in Table II. The network output is converted to MTBF hours by the equation:

$$(\text{output value}) * (\text{Max} - \text{Min}) + \text{Min}.$$

For this training set, Max = 510,000 and Min = 100. This comparison demonstrates the need for training to a much smaller tolerance level.

The 14 and 16 item data sets that used the log of the expected outputs, were able to train to a tolerance of 0.06, the lowest tolerance obtained by any of the networks tested. A comparison of the expected versus the network output MTBF was obtained by presenting a 14 item training set to a network trained with the 16 item data set. The equation for calculating the output value is the same as the previous output value equation with Max = 5.75 and Min = 0.0. This comparison is shown in Table III. This data shows that improving the training has improved some of the outputs, but a lot of improvement is necessary to get predictions that will be reasonably close to the actual data.

Table II Comparison of network output to expected output for the network trained to an error tolerance of 0.1.

Set Number	Expected Output	Network Output
1	57.4	54659
2	29.5	40382
3	2261.7	55169
4	23.2	36303
5	116480	55169
6	2190	27124
7	122640	54149
8	25040	54149
9	1713	31714
10	1140	21516
11	337	55169
12	7280	55679
13	1151	42932
14	46592	55169
Recall Tolerance	0.10	

All of the training conducted, appeared to follow a similar pattern. The output error would follow what appeared to be and exponential decay to zero, or the training tolerance level, if it trained, and to some level other than zero and fluctuate between two or three values when it wouldn't train. An example is the error seen for the 16 item data set with scaled raw

Table III Comparison of expected output to network output for data sets giving the log of the expected output.

Input Number	Expected Output	Network Output
1	57.4	47918
2	29.5	2673
3	2261.7	33516
4	23.2	3303
5	116480	41424
6	2190	1030
7	122640	41976
8	25040	33963
9	1713	2781
10	1140	2466
11	337	37261
12	7280	40879
13	1151	1130
14	46592	41423
Recall Tolerance	0.085	

inputs. An error value of -0.008 was observed for one input pair, and an error of 0.010 for the other input pair in the same set. This was associated with input errors at the same input neurons each time these errors would occur. This indicates that the network is in some sort of limit cycle. This may have been caused by getting caught in a local minimum. The learning rate and the smoothing factor were changed while training to see if the errors would change and

verify local minimum conditions. No change was observed after changing both of these parameters. It appears that the training tolerance limit for this network and data configuration has been found.

F. OBSERVATIONS

This study has proven that training a backpropagation network can take an exorbitant amount of time. The backpropagation network is very sensitive to the inputs it is given. When raw data scaled from 0 to 1 was used, the network could not be trained. When the zeros in the data were changed to 0.2, the network trained very easily to a certain tolerance level. This appears to be caused by the sigmoid transfer functions nonlinearity near zero. When a linear transfer function was used, the network trained with the scaled raw data that contained zeros. Since the goal was to find a nonlinear pattern in the data, this was not a useful transfer function for this problem.

The network responded well to logarithmic changes to the input data. This observation raises the question of whether a logarithmic transfer function might not work better for this data. The Brain Maker software did not have a logarithmic transfer function and did not provide a way to enter a user-defined transfer function, so tests were run with logarithmic changes to the input data itself. This improved the results somewhat but not nearly as much as desired. Therefore, the

conclusion that a logarithmic transfer function would not provide the improvement level desired was made.

The number of inputs to the network were limited to try and bound the problem as quickly as possible. This limit may have created some or all of the training problems experienced. There are other variables that could be entered into the network, and further divisions of the data could be made to try to improve the network accuracy. In fact, after much thought about the problem and current results, it appears there needs to be more inputs to the network if the accuracy of the output values are to be improved.

Adding a second hidden layer to the network may help also. The version of Brain Maker used did not support more than one hidden layer. Therefore, testing this variation of the network could not be accomplished.

Finally, the ability to predict the reliability of a system based on the number of active elements alone does not appear to work based on the analysis performed. Some of the changes mentioned above may make this method useful, but a definite improvement in the training tolerance level is needed.

The process of creating and massaging data inputs and outputs, setting up the backpropagation network, and training the network was becoming more work than generating a classical reliability prediction. This problem led to a search for a simpler and easier to use method of predicting failure rates. Fuzzy-set theory provides a method to simplify inputs and

achieve scalar results, possibly at the expense of some accuracy.

Predicting fielded reliability data is very difficult because one cannot foresee all the possible combinations of factors effecting failure rates. This limitation drove the study to a more basic component level comparison of MIL-HDBK-217E failure predictions to those of the fuzzy model.

The rest of this study examines fuzzy-set theory as a means to provide a simpler method of producing reliability predictions that are accurate enough to be useful to a reliability engineer.

IV. THE FUZZY MODEL

The fuzzy model generates failure rate estimates for MOS, DRAMs. The values generated are for a range of environments listed in Appendix B, Table XII. Before the actual data is reviewed, a discussion of the model function is necessary.

A. FUZZY KNOWLEDGE REPRESENTATIONS

To explain how the model works, we must first look at how the information used by the fuzzy associative memory (FAM) is represented. Fuzzy systems directly encode structured knowledge but in a numerical framework. Associations are created like (Below Average, Low; Almost Low) which means, if the environmental effect on failure rate is below average, and the design effect on failure rate is low, then the failure rate of the component is almost low. The associations are stored in a FAM-rule matrix. Each entry defines a FAM rule. These rules make up the input-output transformations[8].

The fuzzy variables are environment, design, and failure rate. Environment and design are the input variables. Failure rate is the output variable. The fuzzy variable environment is defined by 10 fuzzy-sets as shown in Table IV. The fuzzy variable, design is defined by 5 fuzzy-sets as shown in Table V. The output fuzzy variable, failure rate, is defined by 17 fuzzy-sets as shown in Table VI. These fuzzy-sets were defined based on natural divisions in the overall data shown in Table XIII Appendix B. The histogram in Figure 10 of

Appendix C provides confirmation that 17 groups or fuzzy-sets will cover the data in the table. The matrix created by all the possible fuzzy-set comparisons is shown in Table VII. This matrix defines the fuzzy rules for this FAM network. Each fuzzy rule relates the input fuzzy-sets (environment and design) to an output fuzzy-set (failure rate).

Table IV Linguistic variables for environment effects.

Name	Environmental Possibility of Failure	Name	Environmental Possibility of Failure
LO	Low	SWAA	SomeWhat Above Average
SWL	SomeWhat Low	AA	Above Average
MBA	Much Below Average	MAA	Much Above Average
BA	Below Average	SWH	SomeWhat High
SWBA	SomeWhat Below Average	HI	High

Table V Linguistic variables for design.

Name	Design Failure Rate Affect	Name	Design Failure Rate Affect
LO	Low	AA	Above Average
BA	Below Average	HI	High
AV	Average		

Table VI Linguistic variables for failure rate.

Name	Failure Rate Description	Name	Failure Rate Description
EL	Extremely Low	AAA	Almost Above Average
LO	Low	SWAA	SomeWhat Above Average
SWL	SomeWhat Low	AA	Above Average
AL	Almost Low	EAA	Extremely Above Average
EBA	Extremely Below Average	AH	Almost High
BA	Below Average	SWH	SomeWhat High
SWBA	SomeWhat Below Average	HI	High
ABA	Almost Below Average	EH	Extremely High
AV	Average		

B. SHAPES OF THE FUZZY-SETS

The shapes of the fuzzy-sets define the fuzzy membership functions for the linguistic variables that were defined in the previous section. These shapes can vary depending on the type of input and output variables being represented, and the engineers knowledge and experience. In practice, fuzzy-set values are usually defined as triangles or trapezoids over

Table VII Matrix of FAM rules.**Design Variables**

Environment Variables	LO	EA	AV	AA	HI
LO	EL	LO	EBA	ABA	SWAA
SWL	AL	BA	SWBA	ABA	SWAA
MBA	SWL	BA	ABA	SWAA	AA
EA	AL	SWBA	AV	SWAA	EAA
SWBA	BA	ABA	AAA	AA	EAA
SWAA	SWBA	AV	AA	EAA	AH
AA	ABA	AAA	AA	EAA	AH
MAA	SWAA	AA	EAA	AH	HI
SWH	AA	EAA	AH	SWH	EH
HI	SWH	SWH	SWH	HI	EH

regions on the real line to simplify computations. The fuzzy-sets used for this model are all triangular with maximum y-axis (degree of membership) values of 1.0 and varying base sizes as shown in Figures 2 through 7. The output fuzzy-sets are broken into 3 figures, over the range of possible x-axis values, for clarity.

Figure 2 shows the membership functions or fuzzy-sets for the fuzzy variable, environment. The 0 to 10 range of x-axis values was chosen to make developing the membership functions easier. The ten sets resulted from dividing the data from MIL-HDBK-217E into ten groups based on environment.

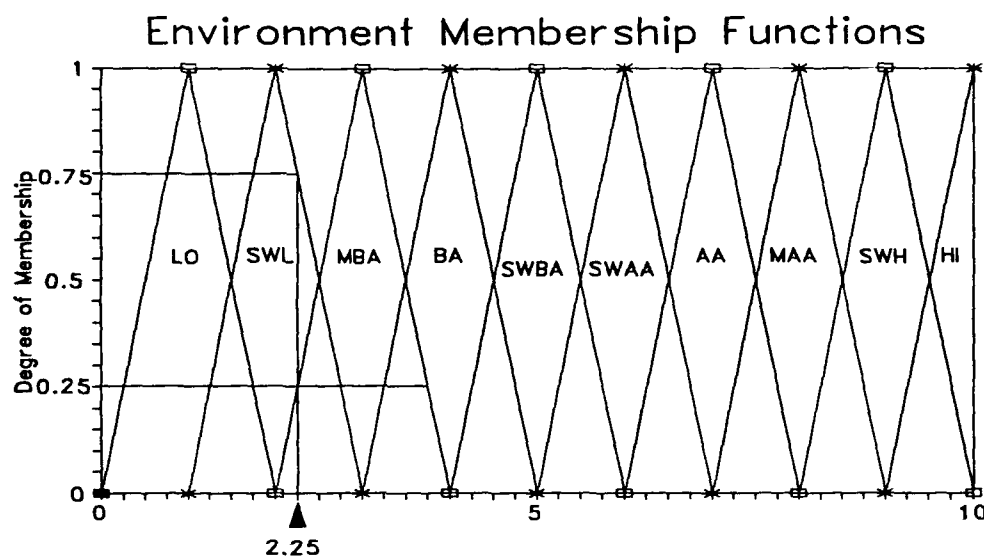


Figure 2 Fuzzy-Sets for Environment.

The membership functions for design are shown in Figure 3. For this model, the design variable describes the number of bits in a dynamic ram chip. For example, average (AV) represents the bit range of 64K to 256K. The x-axis range for the design variable is 0 to 10. Again, the range was chosen to make development of the membership functions easier. Five fuzzy-sets were developed to represent the 4 ranges of bit sizes presented in the table, plus a bit size range that exceeds the table maximum. This bit size is commonly available today, therefore, it was included as an estimated value.

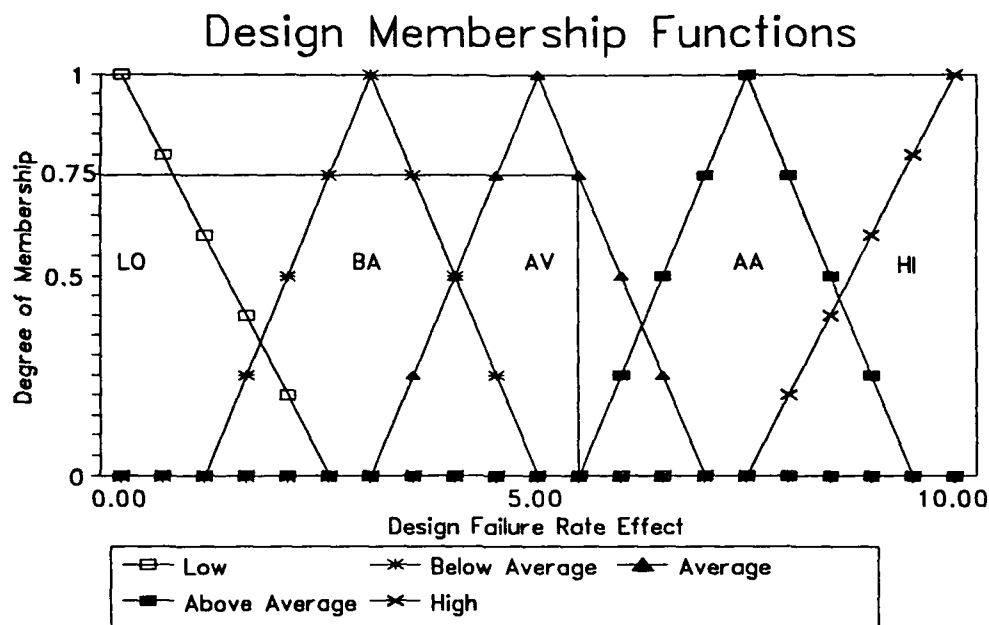


Figure 3 Fuzzy-Sets for Design.

C. BIOFAMS CORRELATION-MINIMUM INFERENCE.

FAM, introduced by Kosko in 1987, is a two layer feed forward, hetroassociative, fuzzy classifier that stores an arbitrary fuzzy spatial pattern pair (A_k, B_k) using fuzzy Hebbian learning, where the k th pattern pair is represented by the fuzzy-sets[8].

Correlation-minimum inference utilizes the max-min composition in order to fire a FAM rule. By using BIOFAMS, we avoid having to keep track of each fuzzy-set value for each FAM rule in a FAM matrix.

With 10 environment, 5 design and 17 failure rate fuzzy-sets, we would have 15 inputs and 17 outputs that would be

tracked in a 15 x 17 matrix. This matrix would be represented by:

$$M=A^T \circ B \quad (1)$$

Where A is a row vector with each element representing the degree of membership in one of the input variables fuzzy-sets and B is a row vector representing the degree of membership in the output variable fuzzy-sets. M represents the matrix created by correlation-minimum encoding where:

$$m_{ij}=\min(a_i, b_j) \quad (2)$$

Each FAM rule would require a 15 x 17 matrix of this type making memory requirements for computer applications quite large.

By employing BIOFAM, the virtual representation scheme is exploited and captures the FAM matrix without physically storing it. A BIOFAM maps system state-variables to the desired output parameters. A BIOFAM can easily accommodate multiple FAM-rule antecedents to map from one fuzzy space to another and therefore, they can greatly simplify the computations involved. A BIOFAM inference procedure activates all of the FAM rules in parallel but at different degrees. Then correlation-minimum is used to activate each FAM rule[15].

For example, take a random access memory (RAM) chip. An experienced engineer qualitatively assesses the failure rate due to the environment it will be used in as somewhat low and assigns it a value of 2.25 on the 0 to 10 scale. This will activate the environment fuzzy-set SWL to degree 0.75, and the

environment fuzzy-set MBA to degree 0.25. Similarly, if the engineer feels the design contribution to failure rate will be average and assigns it a value of 5.5 on the 0 to 10 scale, the design fuzzy-set AV will be activated to degree 0.75. From Table V, the FAM rules activated are:

FAM RULE 8: IF environment = SWL AND design = AV,
THEN failure rate = SWBA

FAM RULE 13: IF environment = MBA AND design = AV,
THEN failure rate = ABA

Figures 2 and 3 on pages 29 and 30 show the fuzzy-sets that activate and the degree of activation for the environment and design variables respectively. Figures 4 through 6 show the failure rate fuzzy-sets in three sections. Figure 4 shows the first 6 failure rate fuzzy-sets. Figure 5 shows the next 6 failure rate fuzzy-sets. And Figure 6 shows the last 5 failure rate fuzzy-sets. Figure 5 shows the failure rate fuzzy-sets for SWBA and ABA. The degree to which the failure rate fuzzy-sets are activated is determined through correlation-minimum encoding[8]. Correlation-minimum encoding compares the degree of activation of the activated fuzzy-sets and selects the minimum value as the degree of activation for the output fuzzy-set as defined earlier.

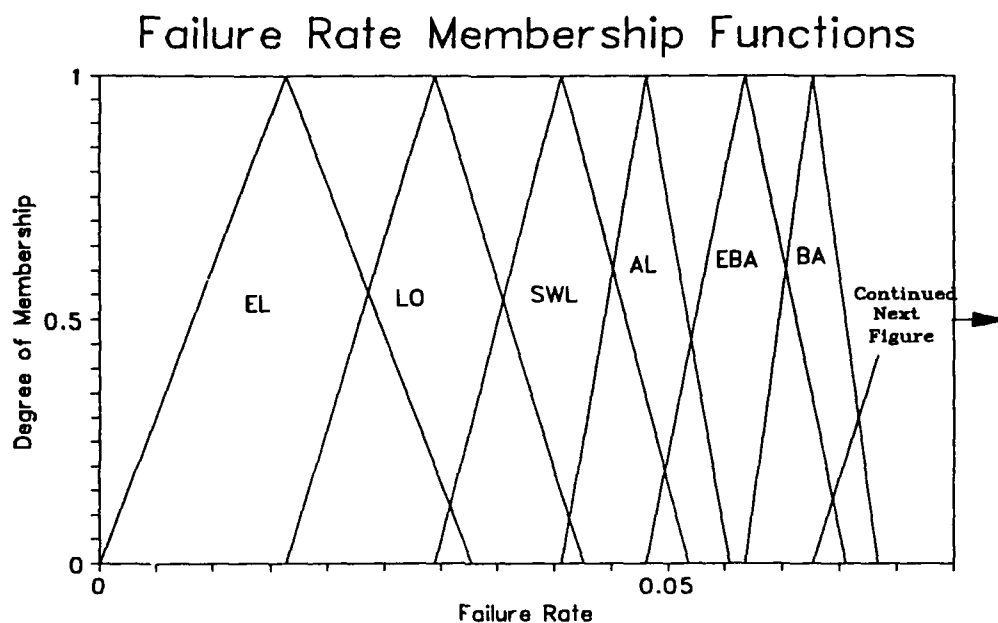


Figure 4 Failure Rate Fuzzy-Sets Part 1.

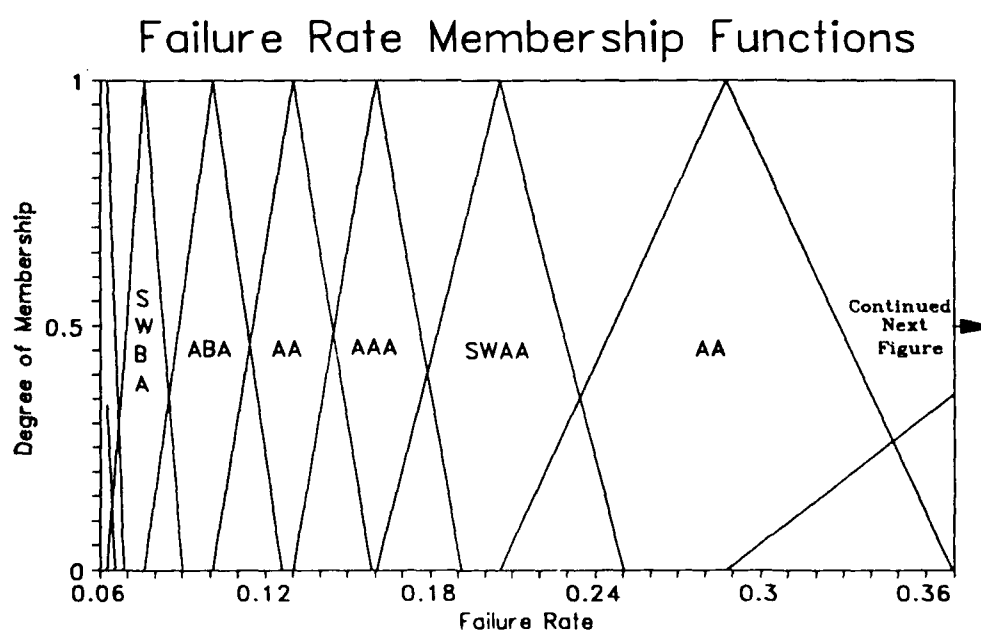


Figure 5 Failure Rate Fuzzy-Sets Part 2.

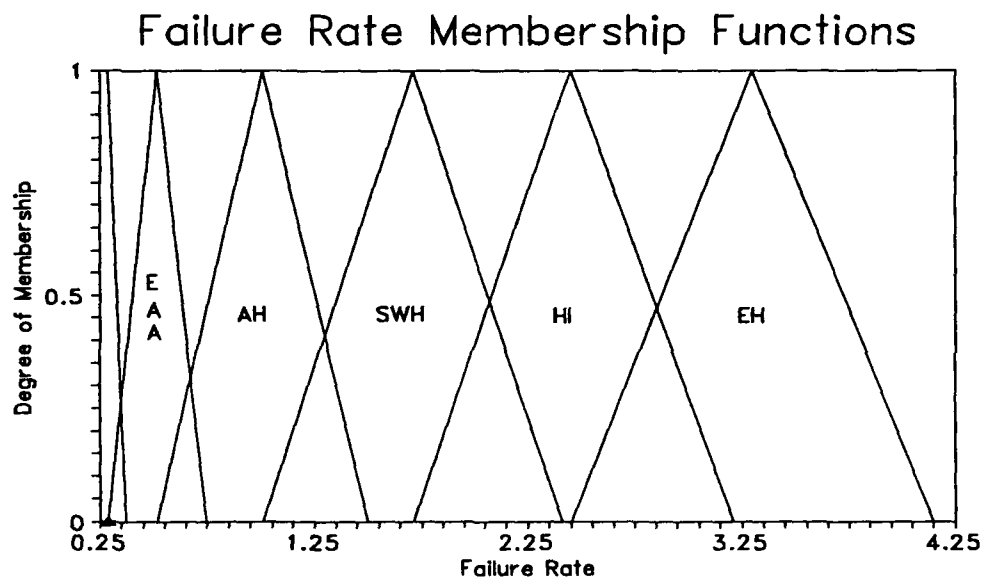


Figure 6 Failure Rate Fuzzy Sets Part 3.

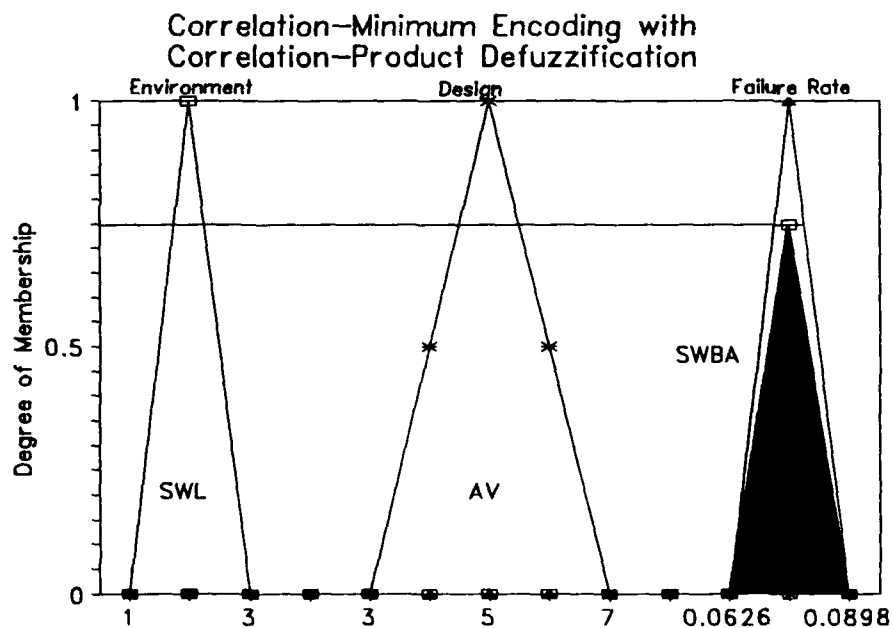


Figure 7 Correlation-Product Encoding of FAM Rule 8.

The method used to determine the degree of activation of the failure rate fuzzy-set SWBA is shown in Figures 7 through 9. To get the degree of activation needed for defuzzification, we need to use a combination of correlation-minimum encoding on the input fuzzy-sets to determine the degree of activation, and correlation-product encoding on the output fuzzy-sets to determine the level of activation and local FAM-rule centroid. Correlation-product encoding is discussed in the next section.

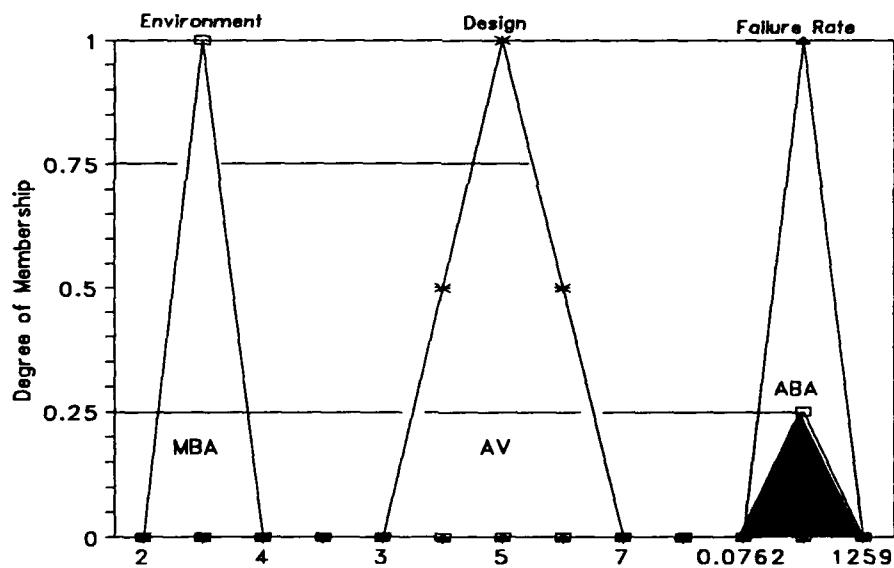


Figure 8 Correlation-Product Encoding of FAM Rule 13.

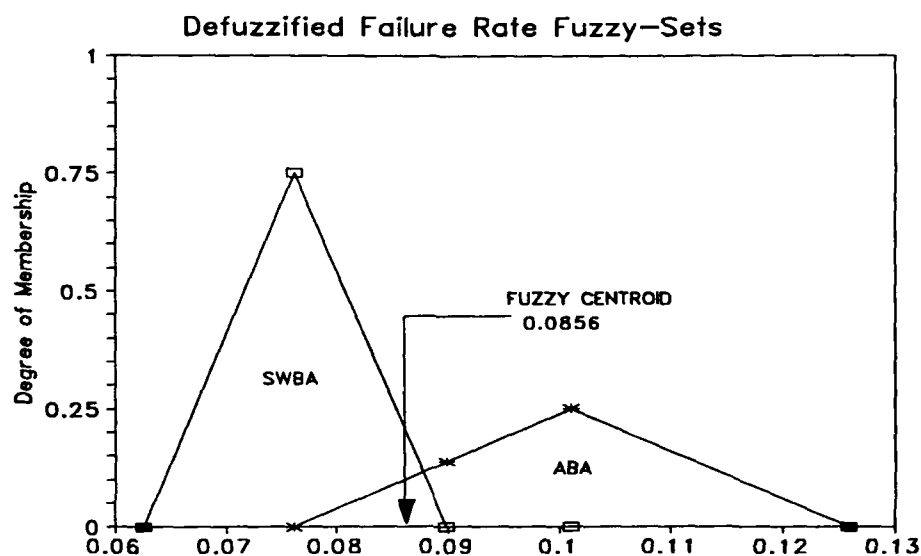


Figure 9 Fuzzy Centroid Determination (Defuzzification).

Using correlation-minimum encoding on the input fuzzy-sets, we get the following:

FAM RULE 8:

$$\min(m_{SWL}^E(2.25), m_{AV}^D(5.5)) \quad (3)$$

$$\min(0.75, 0.75) = 0.75$$

FAM RULE 13:

$$\min(m_{MBA}^E(2.25), m_{AV}^D(5.5)) \quad (4)$$

$$\min(0.25, 0.75) = 0.25$$

Where the superscripts E and D stand for the fuzzy variables environment and design respectively.

As you can see, rules 8 and 13 have been fired in parallel but to different degrees, 0.75 and 0.25 respectively.

D. BIOFAM DEFUZZIFICATION

BIOFAMs can convert the outputs of the rules in the previous section to scalar values through the process of defuzzification. The two most popular defuzzification techniques are maximum membership defuzzification and fuzzy centroid defuzzification.

Maximum membership defuzzification selects the maximum of all the minimum-scaled consequent FAM-rules as follows:

$$m_B(FR_{\max}) = \max_k^i m_B(FR_i) \quad (5)$$

Where FR represents a failure rate fuzzy-set. This equation evaluates each failure rate fuzzy-set, $m_B(FR_i)$, that has fired and chooses the fuzzy-set with the maximum degree of membership $m_B(FR_{\max})$. The degree of membership for the chosen fuzzy-set then becomes the defuzzified scalar value.

Fuzzy centroid defuzzification directly computes the real-valued output as the center of the fuzzy mass C:

$$\bar{y} = \frac{\int_{-\infty}^{\infty} y m_c(y) dy}{\int_{-\infty}^{\infty} m_c(y) dy} \quad (6)$$

The maximum-membership defuzzification scheme has two fundamental problems. First, the mode of the output distribution is not unique. This affects correlation-minimum encoding more than it does correlation-product encoding. Second, the maximum-membership scheme ignores the information on much of the output waveform B or failure rate. Correlation-minimum compounds this problem because B is highly asymmetric, even if it is unimodal. Infinitely many output distributions can share the same mode[8].

The fuzzy centroid scheme is unique and uses all the information in the output distribution B. For fuzzy-sets with symmetric and unimodal distributions, the mode and fuzzy centroid coincide. Therefore, we can replace the integrals in the defuzzification equation with discrete sums[8]. The resulting equation is:

$$\bar{B} = \frac{\sum_{j=1}^P y_j m_B(y_j)}{\sum_{j=1}^P m_B(y_j)} \quad (7)$$

The discrete sum equation requires the computation of the local FAM-rule centroids. This is done through correlation-product inference. Therefore, correlation-product inference must be used to determine the activation levels of the failure

rate fuzzy-sets. This results in a slight change from the correlation-minimum equation as shown:

$$m_{O_i}(y) = w_i m_{B_i}(y_i) \quad (8)$$

Where $m_{O_i}(y)$ is the degree of membership of y in the output fuzzy-set O_i and w_i is the fit value corresponding to the i th output fuzzy-set.

Fuzzy centroid defuzzification is biased toward higher numerical values. For this reason, and because of the problems with the maximum-membership scheme, the fuzzy centroid method of defuzzification is used here.

Now that the defuzzification method has been explained, the example in the previous section is now completed. As mentioned earlier, Figures 7 through 9 on pages 33, 34, and 35, show how the degree of membership of the failure rate fuzzy-sets is determined. They also show, graphically, how fuzzy centroid defuzzification works. The defuzzified scalar result for this example is a failure rate of 0.0856 as shown in Figure 9.

E. MODEL SUMMARY

Figures 7 through 9 on pages 33, 34, and 35 provide the best summary of how the model works.

a. The computer program asks the user to enter a value between 0 and 10 for the environment variable. The program identifies numerical ranges for each linguistic variable as a guide. The program then asks for a numerical value between 0

and 10 for the design variable. Again, numerical ranges are identified for each linguistic variable. These values entered activate one or more fuzzy-sets for each variable, environment and design.

b. Correlation-minimum encoding is used to determine the degree of activation for the output fuzzy-set.

c. The output fuzzy-set(s) is(are) determined from the FAM-rule matrix based on the activated input fuzzy-sets.

d. The degree of activation obtained from correlation-minimum encoding is used to initiate correlation-product encoding with the output fuzzy-sets.

e. Finally the center of mass of the output fuzzy-sets are determined and a defuzzified scalar output is generated. This output represents the failure rate of the component under the specified environment and design effects.

In the next section the results of the model testing are presented. First, some earlier model designs will be discussed. Then a review of current model results is given. Finally, some ways to further develop this concept are suggested.

V. TESTING, RESULTS AND FUTURE WORK

A. EARLY MODEL DESIGN RESULTS

Mathematical manipulation of the data for this model is not difficult, but it requires a large number of calculations. The best way to handle these calculations is through a computer program. The program for this model was written in Turbo Pascal and can be found in Appendix E.

The first model tested consisted of 5 environment fuzzy-sets, 5 design fuzzy-sets, and 5 failure rate fuzzy-sets. This created a 25 cell matrix (25 FAM-rules) that was filled by 5 output fuzzy-sets.

The failure rate values predicted by this model were not close enough to the values in Table XIII of Appendix B to satisfy the author. The differences between the model outputs and the table values were 50 percent or more of the table values.

The fuzzy-set overlap for each of the fuzzy-variables was varied from approximately 50 percent to zero, with little effect. More overlap was better up to about 25 percent, but the desired values were still showing similar differences. The overlap of environment and design fuzzy-sets was varied independently of failure rate also. This met with similar results to those above, the values were not accurate enough and there were large ranges of input values for which the outputs remained constant.

Changing the overlap of the fuzzy-sets changes the number of FAM rules that fire for a given set of inputs. Different numbers of active FAM rules results in different fuzzy centroids and thus different output values for failure rate. Therefore, overlap is one of the key parameters in fuzzy-set development and the model output values obtained.

To improve the output, 9 output (failure rate) fuzzy-sets were created to replace the original 5. This change produced a big improvement in the output data. The output data was now matching the data from Table XIII in Appendix B. Even though the data was much better, there were still some areas that needed improvement. When either one of the variables was assigned a value in the "HI" fuzzy-set, the output would remain at a constant value. Constant values would occur at several input value combinations also. This led to another analysis where the overlap and shape of the fuzzy-sets were varied. After testing a number of iterations varying shapes and overlap, it appeared that changing these parameters would not achieve the desired accuracy.

Attempting to find another way to improve the accuracy of the model, the design variable was modified to 10 fuzzy-sets over an x-axis range of 0 to 4 megabits. The model now had 5 environment fuzzy-sets, 10 design fuzzy-sets, and 9 failure rate fuzzy-sets. So we now have 5×10 or 50 possible FAM-rules and 9 output fuzzy-sets.

This version of the model reduced the range over which values would repeat, but did not eliminate value repetition.

Also, the problem of repeating values when the inputs were in the fuzzy-sets associated with the linguistic value high still remained. This problem indicated that more output fuzzy-sets were needed, especially for the fuzzy-sets associated with the linguistic variable high. Therefore, another redesign of the model occurred. For this version of the model, the inputs and outputs were reevaluated. The input variable, environment, was defined with 10 fuzzy-sets to better define the variation in failure rates due to environment. Once again, the input variable, design, was defined with 5 fuzzy-sets. This was done to keep the model design as simple as possible and because the bit size effects on failure rate have well defined break points which probably do not require as much fuzzy interpretation as the variable, environment. The output variable, failure rate, was defined by 17 fuzzy-sets to reduce the range of values covered by each fuzzy-set and eliminate repeat values.

This version of the network produced values that were very close to the expected values shown in Table XIII of Appendix B. These results are presented in the next section.

B. CURRENT MODEL RESULTS

As mentioned in the previous section, the current model has 10 environment fuzzy-sets, 5 design fuzzy-sets, and 17 failure rate fuzzy-sets. This version of the model produces very good results for all but one of the environments in the base failure rate table, Table XIII in Appendix B. Table XIV

in Appendix D provides a comparison of the model data to the data in Table XIII in Appendix B.

The problem environment is cannon launch (CL). The dynamics of this environment are, on the average, at least an order of magnitude higher than any of the other environments evaluated. Therefore, a less accurate estimate of the failure rate for this environment is accepted in lieu of a more complicated model with more fuzzy-sets.

Although this network design produced much better results the first time it was run, than any of the previous model designs, there was room for improvement. The main problem was instances of repeat values for different input value combinations. Some values were also not matching Table XIII in Appendix B as well as desired. These problems were isolated to values that were derived from combinations of the "AA" and "HI" fuzzy-sets in the design variable. By adjusting the overlap on these and some of the fuzzy-sets for the variable, environment, a very good comparison between the program output and the failure rates was obtained for all environments except cannon launch, as shown in Table XIII of Appendix B.

The values for cannon launch are not as accurate, but they are close to the desired values. The differences are still due mainly to the range covered by the output fuzzy-sets. Adding more fuzzy-sets could improve the output in this range, but it would also reduce the fuzziness of the model, moving it more towards a discrete value association algorithm. Adding more fuzzy-sets also makes it harder to use easily

defined qualitative assessments. The more fuzzy-sets used, the more an expert has to move towards a quantitative assessment of the variable in question. For reasons mentioned in section B of the introduction, this is an undesirable situation from a fuzzy point of view.

C. FOLLOW-ON STUDY SUGGESTIONS

There are three areas that could be addressed in future studies. The first is adding more dimensions or fuzzy variables to the model. Second, adding a FAM-rule learning mechanism that will take a basic set of rules and expand them to fill the FAM-rule matrix. Third, this technique should be expanded to include other component types.

By adding more dimensions or input fuzzy variables to the model, the memory requirements for the computer model are greatly increased because the size of the multi-dimensional matrices that store the FAM information grow rapidly with each dimension. One advantage to the increased size of the network is increased FAM-rule capacity. This can lead to models with much more accuracy and a wider range of failure rate prediction applications. Expanded FAM matrices could also benefit from a FAM-rule learning mechanism. Kosko discusses a Differential Competitive Learning Algorithm for fuzzy controllers that could be applied to this application also[8].

Another direction this study should take is applying this method to other component types. If a network can be developed to predict failure rates for a variety of components, the job

of a reliability engineer during preliminary design could, potentially, be greatly simplified.

These are just a few areas where this study could branch and lead to a potentially useful engineering analysis tool. The results of this study indicate it is a potentially useful substitute method to a handbook generated prediction.

VI. CONCLUSION

The goal of this study was to create a fuzzy system that could predict the failure rate of an electronic component. The component used was a dynamic RAM chip and the data for comparison came from MIL-HDBK-217E. The results are very encouraging.

Over a range of environments and RAM bit sizes, the fuzzy model was able to predict failure rates that were reasonably close to the values in the comparison data. This was possible through careful development of the model fuzzy-sets.

Probably the two most important steps in the fuzzy model development were determining the fuzzy variables, and defining the fuzzy-sets. These elements essentially make up the model. If they are not properly chosen and defined, the model will not perform as desired.

Even after careful selection and definition of the model parameters, some adjustments will probably be necessary before the model produces the desired results. Through judicious adjustments to the fuzzy-sets, one can achieve the required results.

Another interesting observation concerns, the accuracy of the model output. It was very much related to the number of output fuzzy-sets used and requires some amount of judgement to be used during model development. A trade off between accuracy and simplicity is required. Increasing the number of output fuzzy-sets improves accuracy, at the price of increased complexity of the model. Conversely, decreasing the number of

output fuzzy-sets simplifies the model, but may do so at the expense of some accuracy in the model predictions. Understanding these relationships will speed model development.

The results of this study indicate that a fuzzy system could very likely be developed to provide preliminary design failure rate predictions on electronic components. This type of system could potentially be very useful to engineers doing preliminary design failure rate predictions. A fuzzy system can offer an easy to use system that can quickly determine a component failure rate under some qualitatively specified conditions. Eliminating some of the monotonous handbook and/or database searches.

The results of this study warrant continued development of this application of fuzzy-set theory.

APPENDIX A.

BACKPROPAGATION DATA: TABULAR LISTINGS

Table VIII. 70 Item raw data set page 1.

a1	Inputs								Output
	a2	a3	a4	a5	a6	a7	a8	a9	
207	0	0	0	0	3865	11	3630	7506	885.5
0	58	19	0	0	77	0	98	175	164.54
411	1186	5741	48	0	7386	0	33167	40552	5215
79	156	147	0	0	382	108	18722	19212	3352
0	114	309	0	0	423	3	433	859	196.72
317	64	21047	5377	0	26805	0	619	27424	1592.7
0	58	19	0	0	77	0	98	175	8848.4
236	141	900	2	0	1279	0	77	1356	464.85
0	114	309	0	0	423	3	433	859	905
971	237	520	0	0	1728	0	650	2378	504697
555	225	471	0	0	1251	0	488	1739	2190
321	8	16	0	0	345	0	73	418	5840
264	121	2941	0	0	3326	0	261	3587	1084.6
0	58	19	0	0	77	0	98	175	17834
411	1186	5741	48	0	7386	0	33167	40552	7520
264	121	2941	0	0	3326	0	261	3587	1084.6
0	58	19	0	0	77	0	98	175	4381.4
0	114	309	0	0	423	3	433	859	221.39
207	0	0	0	0	3865	11	3630	7506	1752
79	156	147	0	0	382	108	18722	19212	292

Table VIII continued. 70 item raw data set page 2.

a1	a2	a3	a4	Inputs			a7	a8	a9	Output
				a5	a6	a7				
1	0	0	0	0	1	8	97	106	6370.9	
2	0	0	0	0	2	0	205	207	21024	
1	8	0	0	0	9	1	691	701	52560	
2	0	0	0	0	2	0	512	514	105120	
1	8	0	0	0	9	1	691	701	15017	
7	60	147	0	0	214	0	194	408	796.36	
79	156	147	0	0	382	108	18722	19212	61.46	
7	60	147	0	0	214	0	194	408	5840	
0	0	0	0	0	0	0	29	29	20240	
0	0	0	0	0	0	0	10	10	210240	
0	114	309	0	0	423	3	433	859	215	
317	64	21047	5377	0	26805	0	619	27424	651	
0	114	309	0	0	423	3	433	859	143.73	
0	58	19	0	0	77	0	98	175	3138.2	
0	114	309	0	0	423	3	433	859	192.93	
0	114	309	0	0	423	3	433	859	331.6	
236	141	900	2	0	1279	0	77	1356	8760	
0	114	309	0	0	423	3	433	859	499.88	
0	58	19	0	0	77	0	98	175	13171	
0	58	19	0	0	77	0	98	175	1606.5	
0	114	309	0	0	423	3	433	859	601.84	
0	114	309	0	0	423	3	433	859	444.99	
0	2	5	0	0	7	0	23	30	18222	
0	2	5	0	0	7	0	23	30	10362	
0	58	19	0	0	77	0	98	175	1399.8	

Table VIII continued. 70 item raw data set page 3.

a1	a2	a3	a4	Inputs			a6	a7	a8	a9	Output
				a5	a5	a5					
0	58	19	0	0	0	0	77	0	98	175	120.87
0	2	5	0	0	0	0	7	0	23	30	1289.7
317	64	21047	5377	0	26805	0	26805	0	619	27424	542.33
0	2	5	0	0	0	0	7	0	23	30	10009
0	58	19	0	0	0	0	77	0	98	175	3524.2
264	121	2941	0	0	3326	0	3326	0	261	3587	25281
0	58	19	0	0	77	0	77	0	98	175	3694
316	123	2379	48	0	2866	0	2866	0	253	3119	25281
0	58	19	0	0	77	0	77	0	98	175	2475
0	58	19	0	0	77	0	77	0	98	175	6395
0	114	309	0	0	423	3	423	3	433	859	343.5
0	114	309	0	0	423	3	423	3	433	859	1519.7
0	58	19	0	0	77	0	77	0	98	175	2299.8
0	58	19	0	0	77	0	77	0	98	175	1664
0	2	5	0	0	7	0	7	0	23	30	2029.9
0	58	19	0	0	77	0	77	0	98	175	1723
0	58	19	0	0	77	0	77	0	98	175	1423.9
0	114	309	0	0	423	3	423	3	433	859	397.85
0	58	19	0	0	77	0	77	0	98	175	2290
0	2	5	0	0	7	0	7	0	23	30	951.22
0	58	19	0	0	77	0	77	0	98	175	2221.2
0	114	309	0	0	423	3	423	3	433	859	394.21
0	2	5	0	0	7	0	7	0	23	30	2466.5
0	114	309	0	0	423	3	423	3	433	859	210.06
0	58	19	0	0	77	0	77	0	98	175	5223.6

Table IX. 16 item raw data set.

a1	Inputs								Output
	a2	a3	a4	a5	a6	a7	a8	a9	
207	0.2	0.2	0.2	0.2	3865	11	3630	7506	1504
316	123	2379	48	0.2	2866	0.2	253	3119	25281
317	64	21047	5377	0.2	26805	0.2	619	27424	1092.1
236	141	900	2	0.2	1279	0.2	77	1356	2308.2
555	225	471	0.2	0.2	1251	0.2	488	1739	2190
321	8	16	0.2	0.2	345	0.2	73	418	5840
264	121	2941	0.2	0.2	3326	0.2	261	3587	4004.6
1	0.2	0.2	0.2	0.2	1	8	97	106	6370.9
2	0.2	0.2	0.2	0.2	2	0.2	205	207	21024
1	8	0.2	0.2	0.2	9	1	691	701	52560
411	1186	5741	48	0.2	7386	0.2	33167	40552	21134
2	0.2	0.2	0.2	0.2	2	0.2	512	514	105120
1	8	0.2	0.2	0.2	9	1	691	701	15017
7	60	147	0.2	0.2	214	0.2	194	408	1401.6
79	156	147	0.2	0.2	382	108	18722	19212	214.37
971	237	520	0.2	0.2	1728	0.2	650	2378	504696

Table X. 14 Item data set with 100,000 and higher MTBFs removed.

a1	Inputs								Output
	a2	a3	a4	a5	a6	a7	a8	a9	
207	0.2	0.2	0.2	0.2	3865	11	3630	7506	1504
316	123	2379	48	0.2	2866	0.2	253	3119	25281
317	64	21047	5377	0.2	26805	0.2	619	27424	1092.1
236	141	900	2	0.2	1279	0.2	77	1356	2308.2
555	225	471	0.2	0.2	1251	0.2	488	1739	2190
321	8	16	0.2	0.2	345	0.2	73	418	5840
264	121	2941	0.2	0.2	3326	0.2	261	3587	4004.6
1	0.2	0.2	0.2	0.2	1	8	97	106	6370.9
2	0.2	0.2	0.2	0.2	2	0.2	205	207	21024
1	8	0.2	0.2	0.2	9	1	691	701	52560
411	1186	5741	48	0.2	7386	0.2	33167	40552	21134
1	8	0.2	0.2	0.2	9	1	691	701	15017
7	60	147	0.2	0.2	214	0.2	194	408	1401.6
79	156	147	0.2	0.2	382	108	18722	19212	214.37

Table XI. 16 Item data set with zero = 0.2 and output is log of desired number.

a1	a2	a3	a4	Inputs					a7	a8	a9	Output
				a5	a6	a7	a8	a9				
207	0	0	0	0	3865	11	3630	7506				3.177
316	123	2379	48	0	2866	0	253	3119				4.403
317	64	21047	5377	0	26805	0	619	27424				3.038
236	141	900	2	0	1279	0	77	1356				3.363
555	225	471	0	0	1251	0	488	1739				3.34
321	8	16	0	0	345	0	73	418				3.766
264	121	2941	0	0	3326	0	261	3587				3.603
1	0	0	0	0	1	8	97	106				3.804
2	0	0	0	0	2	0	205	207				4.323
1	8	0	0	0	9	1	691	701				4.721
411	1186	5741	48	0	7386	0	33167	40552				4.325
2	0	0	0	0	2	0	512	514				5.022
1	8	0	0	0	9	1	691	701				4.177
7	60	147	0	0	214	0	194	408				3.147
79	156	147	0	0	382	108	18722	19212				2.331
971	237	520	0	0	1728	0	650	2378				5.703

APPENDIX B.

MIL-HDBK-217E DATA

Table XII. MIL-HDBK-217E Environment abbreviation definitions.

SYMBOL	ENVIRONMENT
GB	Ground, Benign
GMS	Ground, Missile Silo
GF	Ground, Fixed
GM	Ground, Mobile
SF	Space, Flight
MP	Manpack
NS	Naval, Sheltered
NU	Naval, Unsheltered
NUU	Naval, Undersea, Unsheltered
NSB	Naval, Submarine
NH	Naval, Hydrofoil
AIC	Airborne, Inhabited, Cargo
AIT	Airborne, Inhabited, Trainer
AIB	Airborne, Inhabited, Bomber
AIA	Airborne, Inhabited, Attack
AIF	Airborne, Inhabited, Fighter
AUC	Airborne, Uninhabited, Cargo
AUT	Airborne, Uninhabited, Trainer
AUB	Airborne, Uninhabited, Bomber
AUA	Airborne, Uninhabited, Attack
AUF	Airborne, Uninhabited, Fighter
ARW	Airborne, Rotary Winged
ML	Missile, Launch
USL	Undersea, Launch
MFF	Missile, Free Flight
MFA	Missile, Flight, Airbreathing
CL	Cannon, Launch

APPENDIX C.

HISTOGRAM OF TABLE XIII DATA

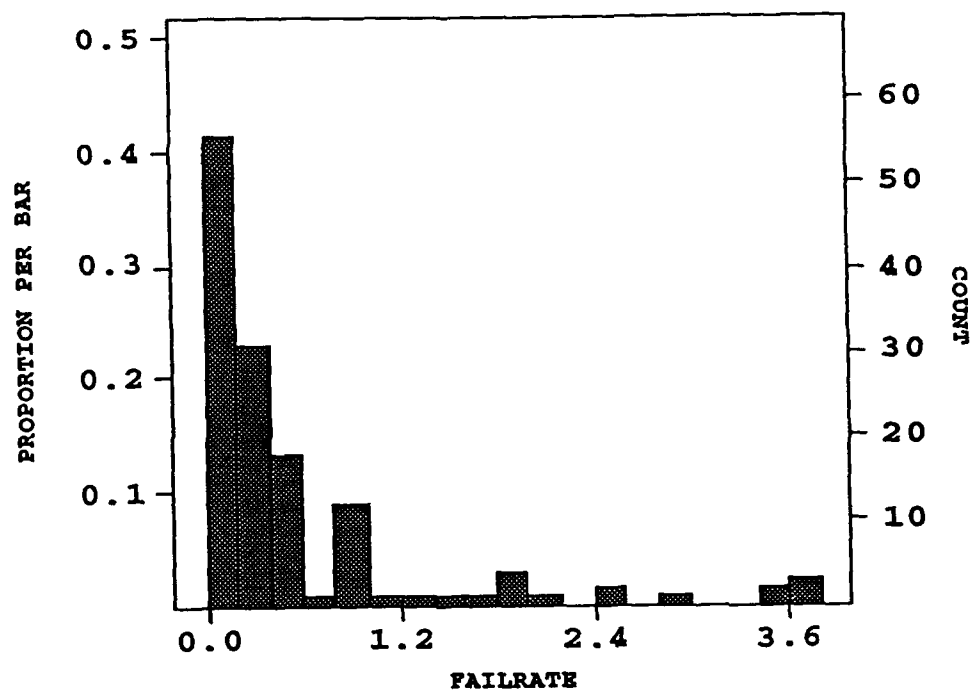


Figure 10 Histogram of Table XIII Data.

APPENDIX D.

COMPARISON OF MODEL DATA TO MIL-HDBK-217E DATA

Table XIV. Comparison of model failure rate predictions to Table XIII.
Failure Rates for MOS Dynamic RAMS (failures/ 10^6 hrs.)

Failure Rates for MOS Dynamic RAMS (failures/10 ⁶ hrs.)										
DEVICE DESCRIPTION		Linguistic Variable								
NO. OF BITS	TECHNOLOGY	LO	GB	GMS	SWL	NUU	MBA	GF	MP	MBA
LO	MOS	0.0163	0.0155	0.0172	0.0481	0.0470	0.0407	0.0395	0.0418	
BA	MOS	0.0294	0.0292	0.0313	0.0626	0.0636	0.0626	0.0671	0.0653	
AV	MOS	0.0568	0.0556	0.0580	0.0762	0.0825	0.1011	0.1163	0.1036	
AA	MOS	0.1011	0.1086	0.1113	0.1011	0.1202	0.2322	0.2149	0.1803	
HI	MOS	0.1858			0.2053		0.2874			
		BA	NSB	NS	SWBA	GM	MFA	SWAA	AIC	AIA
LO	MOS	0.0481	0.0491	0.0491	0.0626	0.0581	0.0657	0.0762	0.0699	0.0794
BA	MOS	0.0762	0.0789	0.0789	0.1011	0.0960	0.1054	0.1299	0.1277	0.1396
AV	MOS	0.1299	0.1293	0.1293	0.1604	0.1620	0.1724	0.2374	0.2377	0.2507
AA	MOS	0.2053	0.2303	0.2303	0.2874	0.2943	0.3066	0.5169	0.4576	0.4729
HI	MOS	0.5169			0.5169			1.0086		
		AA	AIB	ML	MAA	NU	SWH	AUC	AUF	SWH
LO	MOS	0.1011	0.0858	0.1366	0.2053	0.1823	0.2874	0.2501	0.2882	
BA	MOS	0.1604	0.1474	0.2105	0.2874	0.3372	0.5169	0.4857	0.5330	
AV	MOS	0.2874	0.2593	0.3286	0.5169	0.6338	1.0086	0.9500	1.0020	
AA	MOS	0.5169	0.4832	0.5651	1.0086	1.2271	1.7110	1.8787	1.9401	
HI	MOS	1.0086			2.4500		3.3000			
		HI	CL	HI	Design (Bit Size) Linguistic Equivalents					
LO	MOS	1.7110	1.4209		LO = <= 16K					
BA	MOS	1.7110	1.7827		BA = > 16K - 64K					
AV	MOS	1.7110	2.0010		AV = > 64K - 256K					
AA	MOS	2.4500	2.4400		AA = > 256K - 1M					
HI	MOS	3.3000			HI = > 1M - 4M					

Shaded areas indicate model predictions.

Unshaded areas to the right of each shaded block are Table XIII comparisons.

APPENDIX E.

PASCAL PROGRAM FOR FUZZY FAILURE RATE PREDICTION

```

PROGRAM FUZZY_FAILURE_RATE_PREDICTOR (Input, Output,
    Factorlist, Afunctions, Bfunctions, Rulelist, Lambda);
USES Crt;
CONST Maxfactor      = 2;
      Factor1_size   = 10;
      Factor2_size   = 5;
      Components     = 1;
      Result_size    = 17;

TYPE Component_Range = 1..Components;
      Factor_Range   = 1..Maxfactor;
      Ax1_Range      = 1..Factor1_size;
      Ax2_Range      = 1..Factor2_size;
      Ex_Range       = 1..Result_size;
      Factor_type     = ARRAY [1..Maxfactor] OF Real;
      Level_type      = ARRAY [Ax1_Range, Ax2_Range] OF
                          Boolean;
      Rule_type       = ARRAY [Component_Range, Ax2_Range,
                              Ax1_Range] OF Integer;
      Ax_type         = ARRAY [Factor_Range, Ax1_Range] OF
                          Real;
      Bx_type         = ARRAY [Factor_Range, Ax1_Range] OF
                          Real;
      Cx_type         = ARRAY [Factor_Range, Ax1_Range] OF
                          Real;
      Dx_type         = ARRAY [Factor_Range, Ax1_Range] OF
                          Real;
      Ex_type         = ARRAY [Component_Range, Ex_Range] OF
                          Real;
      Fx_type         = ARRAY [Component_Range, Ex_Range] OF
                          Real;
      Gx_type         = ARRAY [Component_Range, Ex_Range] OF
                          Real;
      Hx_type         = ARRAY [Component_Range, Ex_Range] OF
                          Real;
      Ay_type         = ARRAY [Factor_Range, Ax1_Range] OF
                          Real;
      By_type         = ARRAY [Factor_Range, Ax1_Range] OF
                          Real;
      Cy_type         = ARRAY [Factor_Range, Ax1_Range] OF
                          Real;
      Dy_type         = ARRAY [Factor_Range, Ax1_Range] OF
                          Real;
      Ey_type         = ARRAY [Component_Range, Ex_Range] OF
                          Real;
      Fy_type         = ARRAY [Component_Range, Ex_Range] OF
                          Real;
      Gy_type         = ARRAY [Component_Range, Ex_Range] OF
                          Real;
      Hy_type         = ARRAY [Component_Range, Ex_Range] OF
                          Real;

```



```

M_type      = ARRAY [Factor_Range, Ax1_Range] OF
              Real;
C_type      = ARRAY [Component_Range, Ex_Range] OF
              Real;
L_type      = ARRAY [Component_Range, Ex_Range] OF
              Real;

VAR  Ax: Ax_type;
     Bx: Bx_type;
     Cx: Cx_type;
     Dx: Dx_type;
     Ay: Ay_type;
     By: By_type;
     Cy: Cy_type;
     Dy: Dy_type;
     Ex: Ex_type;
     Ey: Ey_type;
     Fx: Fx_type;
     Fy: Fy_type;
     Gx: Gx_type;
     Gy: Gy_type;
     Hx: Hx_type;
     Hy: Hy_type;
     M:  M_type;
     C:  C_type;
     L:  L_type;
     Lambda: Text;

(*****
FUNCTION LowerSlope (VAR X1,Y1,X2,Y2: Real): Real;
BEGIN
  IF ((X1=X2) OR (Y1=Y2))
    THEN LowerSlope := 0
    ELSE LowerSlope := Y2/(X2-X1);
END;
(*****
(*****
FUNCTION UpperSlope (VAR X3,Y3,X4,Y4: Real): Real;
BEGIN
  IF ((X3=X4) OR (Y3=Y4))
    THEN UpperSlope := 0
    ELSE UpperSlope := Y3/(X3-X4);
END;
(*****
(*****
FUNCTION LowerInt (VAR X1,Y1,X2,Y2: Real): Real;
BEGIN
  IF (Y1 = Y2)
    THEN LowerInt := Y1
    ELSE IF (X1=X2)
      THEN LowerInt := 100000
      ELSE LowerInt := -(Y2/(X2-X1)) * X1;
END;

```

```

(*****)
(*****)
FUNCTION UpperInt (VAR X3,Y3,X4,Y4: Real): Real;
BEGIN
    IF (Y3=Y4)
    THEN UpperInt := Y3
    ELSE IF (X3=X4)
    THEN UpperInt := 100000
    ELSE UpperInt := -(Y3/(X3-X4)) * X4;
    END;
(*****)
(*****)
FUNCTION Max (VAR q, r: Real): Real;
BEGIN
    IF q >= r
    THEN
        Max := q
    ELSE
        Max := q;
    END;
(*****)
(*****)
FUNCTION Min (VAR s,t: Real): Real;
BEGIN
    IF s > t
    THEN Min := t
    ELSE Min := s;
    END;
(*****)
(*****)
FUNCTION Trap_area (VAR X1,Y1,X2,Y2,X3,Y3,X4,Y4: Real):
    Real;
BEGIN
    Trap_area := ((X2-X1)*(Y2-Y1))+((Y2-Y1)*(X3-X2));
    END;
(*****)

{*****}
PROCEDURE Get_Failure_Rate_Factors (VAR fact: Factor_type);

    VAR    Maxfact,i, b: Integer;
           factor: ARRAY [1..Maxfactor] OF String[22];
           Factorlist: Text;

    BEGIN
        ASSIGN (Factorlist, 'factors.lst');
        RESET (Factorlist);
        BEGIN {Getting the list of factors to consider.}
            FOR i := 1 TO Maxfactor DO
                BEGIN
                    Readln(Factorlist, factor[i]);
                END;
            END;
        END;

```

```

{START Getting the factor values.}
FOR b := 1 TO Maxfactor DO
  IF (b = 1)
    THEN BEGIN
      Writeln ('Enter a ',factor[b],' factor value
               from 0.000 and 10.000. ');
      Writeln ('0.000 - 2.000 = Low');
      Writeln ('1.000 - 3.000 = Somewhat Low');
      Writeln ('2.000 - 4.000 = Much Below Average');
      Writeln ('3.000 - 5.000 = Below Average');
      Writeln ('4.000 - 6.000 = Somewhat Below
               Average');
      Writeln ('5.000 - 7.000 = Somewhat Above
               Average');
      Writeln ('6.000 - 8.000 = Above Average');
      Writeln ('7.000 - 9.000 = Much Above Average');
      Writeln ('8.000 - 10.000 = Somewhat Highe');
      Writeln ('9.000 - 10.000 = High');
      Readln (fact[b]);
      Write('Fact ',b,' = ',fact[b]:2:3);
      Writeln;
    END
  ELSE BEGIN
      Writeln ('Enter a ',factor[b],' factor value
               from 0.000 and 10.000. ');
      Writeln ('0.000 - 2.000 = Little');
      Writeln ('2.000 - 4.000 = Below Average');
      Writeln ('4.000 - 6.000 = Average');
      Writeln ('6.000 - 8.000 = Above Average');
      Writeln ('8.000 - 10.000 = Very');
      Readln (fact[b]);
      Write('Fact ',b,' = ',fact[b]:2:3);
      Writeln;
    END;
  Close (Factorlist);
END;

{*****}
PROCEDURE Define_Membership_Functions (VAR Ax: Ax_type;
                                       VAR Bx: Bx_type;
                                       VAR Cx: Cx_type;
                                       VAR Dx: Dx_type;
                                       VAR Ay: Ay_type;
                                       VAR By: By_type;
                                       VAR Cy: Cy_type;
                                       VAR Dy: Dy_type);

VAR i,j,Factor_size: Integer;
    Afunctions: Text;

```

```

BEGIN
  Assign (Afunctions, 'Afunct.dat');
  Reset  (Afunctions);

  FOR i := 1 TO Maxfactor DO
    BEGIN
      Readln (Afunctions);
      IF (i = 1)
        THEN Factor_size := Factor1_size
        ELSE Factor_size := Factor2_size;
      FOR j := 1 TO Factor_size DO
        BEGIN
          Read (Afunctions, Ax[i,j]);
          Read (Afunctions, Ay[i,j]);
          Read (Afunctions, Bx[i,j]);
          Read (Afunctions, By[i,j]);
          Read (Afunctions, Cx[i,j]);
          Read (Afunctions, Cy[i,j]);
          Read (Afunctions, Dx[i,j]);
          Read (Afunctions, Dy[i,j]);
          Readln (Afunctions);
        END; {FOR}
      END; {FOR}
      Close (Afunctions);
    END; {Procedure: Define_Membership_Functions}
  (*****)
  PROCEDURE Calculate_Input_Membership_Functions
    (VAR M: M_type;
     VAR Ax: Ax_type;
     VAR Bx: Bx_type;
     VAR Cx: Cx_type;
     VAR Dx: Dx_type;
     VAR Ay: Ay_type;
     VAR By: By_type;
     VAR Cy: Cy_type;
     VAR Dy: Dy_type;
     VAR fact: Factor_type);

  VAR i,j,k,Factor_size: Integer;

  BEGIN
    FOR i := 1 TO Maxfactor DO
      BEGIN
        IF (i = 1)
          THEN Factor_size := Factor1_size
          ELSE Factor_size := Factor2_size;
        FOR j := 1 TO Factor_size DO
          BEGIN
            IF (fact[i] <= Ax[i,j])
              THEN M[i,j] := Ay[i,j]
              ELSE IF ((fact[i] > Ax[i,j]) AND (fact[i] <=
                Bx[i,j]))

```

```

THEN M[i,j] :=
  ((fact[i]*LowerSlope(Ax[i,j],Ay[i,j],
    Bx[i,j], By[i,j])) + LowerInt(Ax[i,j],
    Ay[i,j], Bx[i,j],By[i,j]))
ELSE IF ((fact[i] > Bx[i,j]) AND (fact[i] <=
  Cx[i,j]))
  THEN M[i,j] :=
    ((fact[i]*LowerSlope(Bx[i,j],
    By[i,j],Cx[i,j],Cy[i,j]))+
    LowerInt(Bx[i,j],By[i,j],
    Cx[i,j],Cy[i,j]))
  ELSE IF ((fact[i] > Cx[i,j]) AND (fact[i]
    <= Dx[i,j]))
    THEN M[i,j] :=
      ((fact[i]*UpperSlope(Cx[i,j],
      Cy[i,j],Dx[i,j],Dy[i,j]))
      +UpperInt(Cx[i,j],Cy[i,j],
      Dx[i,j],Dy[i,j]))
    ELSE IF (fact[i] > Dx[i,j])
      THEN M[i,j] := Dy[i,j];
  END; {FOR j}
END; {FOR i}
END;

{*****}
PROCEDURE Read_Component_Membership_Functions
  (VAR Ex: Ex_type;
   VAR Ey: Ey_type;
   VAR Fx: Fx_type;
   VAR Fy: Fy_type;
   VAR Gx: Gx_type;
   VAR Gy: Gy_type;
   VAR Hx: Hx_type;
   VAR Hy: Hy_type);

VAR i,j: Integer;
    Bfunctions: Text;

BEGIN
  ASSIGN (Bfunctions, 'bfunct.dat');
  RESET (Bfunctions);

  FOR i := 1 TO Components DO
    BEGIN
      Readln (Bfunctions);
      FOR j := 1 TO Result_size DO
        BEGIN
          Read (Bfunctions, Ex[i,j]);
          Read (Bfunctions, Ey[i,j]);
          Read (Bfunctions, Fx[i,j]);
          Read (Bfunctions, Fy[i,j]);
          Read (Bfunctions, Gx[i,j]);
          Read (Bfunctions, Gy[i,j]);

```

```

        Read (Bfunctions, Hx[i,j]);
        Read (Bfunctions, Hy[i,j]);
        Readln (Bfunctions);
    END; {FOR j}
END; {FOR i}
Close (Bfunctions);
END;      {Procedure: Read_Component_Membership_Functions}

(*****)
PROCEDURE Calculate_Areas_and_Centers (VAR Ex: Ex_type;
                                       VAR Ey: Ey_type;
                                       VAR Fx: Fx_type;
                                       VAR Fy: Fy_type;
                                       VAR Gx: Gx_type;
                                       VAR Gy: Gy_type;
                                       VAR Hx: Hx_type;
                                       VAR Hy: Hy_type;
                                       VAR C:  C_type;
                                       VAR L:  L_type);

VAR i,j,k: Integer;

BEGIN
    FOR i := 1 TO Components DO;
        BEGIN
            FOR j := 1 TO Result_size DO
                BEGIN
                    L[i,j] := Trap_area(Ex[i,j],Ey[i,j],
                                       Fx[i,j],Fy[i,j],Gx[i,j],Gy[i,j],
                                       Hx[i,j],Hy[i,j]);
                    IF (Fx[i,j] = Gx[i,j])
                        THEN C[i,j] := Fx[i,j]
                        ELSE C[i,j] := (Ex[i,j] + Hx[i,j])/2;
                END;
                Writeln;
            END;
        END;      {Calculate_Areas_and_Centers}

(*****)
PROCEDURE Read_Rules (VAR Rule: Rule_type);

VAR i,j,k: Integer;
    Rulelist: Text;

BEGIN
    ASSIGN (Rulelist, 'Rules.lst');
    RESET (Rulelist);
    Readln(Rulelist);
    FOR k := 1 TO Components DO
        BEGIN
            FOR i := 1 TO Factor2_size DO
                BEGIN

```

```

        FOR j := 1 TO Factor1_size DO
            BEGIN
                Read(Rulelist, Rule[k,i,j]);
                Write('Rule[' ,k,i,j, ']=' ,Rule[k,i,j], ' ');
            END;
            Readln(Rulelist);
        END;
    END;
    Close(Rulelist);
END;

{*****}
PROCEDURE Calculate_Failure_Rate_Factor
    (VAR Rule: Rule_type;
     VAR M: M_type;
     VAR L: L_type;
     VAR C: C_type;
     VAR Lambda: Text);

VAR i,j,k,n,p: Integer;
    Nochange: Boolean;
    Degree, RuleArea, Center, Centroid, Num, Denom, X2, X3: Real;

BEGIN
    Num := 0;
    Denom := 0;
    FOR k := 1 TO Components DO
        BEGIN
            FOR i := 1 TO Factor2_size DO
                BEGIN
                    FOR j := 1 TO Factor1_size DO
                        BEGIN
                            IF ((M[1,j] > 0) AND (M[2,i] > 0))
                                THEN
                                    BEGIN
                                        n := Rule[k,i,j];
                                        Degree := Min(M[1,j], M[2,i]);
                                        RuleArea := L[k,n];
                                        Center := C[k,n];
                                        Num := Num + Degree*
                                            RuleArea*Center;
                                        Denom := Denom + Degree*
                                            RuleArea;
                                    END
                                ELSE Nochange := True;
                            END; {FOR j}
                        END; {FOR i}
                    END; {FOR k}
                END;
            Writeln;
            Writeln('Num=', Num:2:4);
            Writeln('Denom=', Denom:2:4);
            Writeln('Degree=', Degree:2:4);
            Centroid := Num/Denom;
            Writeln('The Failure Rate Factor is: ', Centroid:2:4);
        
```

```

Write(Lambda, 'Degree=', Degree:2:4, ' ');
Write (Lambda, 'Failure Rate = ', Centroid:2:4);
Writeln;
END;

{*****}
PROCEDURE Create_Output_File (VAR fact: Factor_type;
                             VAR Lambda: TEXT);
VAR Level: Level_type;
    Rule: Rule_type;

CONST q = 1;
      r = 2;
      stepq = 1;
      stepr = 1;

BEGIN
  Assign (Lambda, 'LambdaG.DAT');
  Append (Lambda);
  fact[q] := 0;

  While fact[q] < 10 DO
    BEGIN
      fact[q] := fact[q] + stepq;
      fact[r] := 0;
      While fact[r] < 10 DO
        BEGIN
          fact[r] := fact[r] + stepr;
          Write (Lambda, 'fact['',q,']=', fact[q]:2:2, '
            ', 'fact['',r,']=', fact[r]:4:2, ' ');
          Define_Membership_Functions
            (Ax, Bx, Cx, Dx, Ay, By, Cy, Dy);
          Read_Component_Membership_Functions
            (Ex, Ey, Fx, Fy, Gx, Gy, Hx, Hy);
          Calculate_Input_Membership_Functions
            (M, Ax, Bx, Cx, Dx, Ay, By, Cy, Dy, fact);
          Calculate_Areas_and_Centers
            (Ex, Ey, Fx, Fy, Gx, Gy, Hx, Hy, C, L);
          Read_Rules(Rule);
          Calculate_Failure_Rate_Factor
            (Rule, M, L, C, Lambda);
          Writeln(Lambda);
        END; {While r}
      Writeln(Lambda);
    END; {While q}
  Close (Lambda);
END; {Create_Output_File}

```



```

{*****}
VAR {Level: Level_type;}
    Rule: Rule_type;
    fact: Factor_type;

BEGIN {Determining the factor values.}
{      Create_Output_File(fact,Lambda);    }
  Assign (Lambda,'Lambdag.dat');
  Append (Lambda);
  Get_Failure_Rate_Factors (fact);
  Define_Membership_Functions (Ax,Bx,Cx,Dx,Ay,By,Cy,Dy);
  Read_Component_Membership_Functions
    (Ex,Ey,Fx,Fy,Gx,Gy,Hx,Hy);
  Calculate_Input_Membership_Functions
    (M,Ax,Bx,Cx,Dx,Ay,By,Cy,Dy,fact);
  Calculate_Areas_and_Centers
    (Ex,Ey,Fx,Fy,Gx,Gy,Hx,Hy,C,L);
  Read_Rules(Rule);
  Calculate_Failure_Rate_Factor (Rule,M,L,C,lambda);
END.

```

APPENDIX F.

DATA FILES USED BY THE MODEL

X, Y coordinate definitions of membership functions

Environment Membership Function Definitions (AFUNCT.DAT)

	Ax	Ay	Bx	By	Cx	Cy	Dx	Dy
LO	00.00	0.00	01.00	1.00	01.00	1.00	02.00	0.00
SWL	01.00	0.00	02.00	1.00	02.00	1.00	03.00	0.00
MBA	02.00	0.00	03.00	1.00	03.00	1.00	04.00	0.00
BA	03.00	0.00	04.00	1.00	04.00	1.00	05.00	0.00
SWBA	04.00	0.00	05.00	1.00	05.00	1.00	06.00	0.00
SWAA	05.00	0.00	06.00	1.00	06.00	1.00	07.00	0.00
AA	06.00	0.00	07.00	1.00	07.00	1.00	08.00	0.00
MAA	07.00	0.00	08.00	1.00	08.00	1.00	09.00	0.00
SWH	08.00	0.00	09.00	1.00	09.00	1.00	10.00	0.00
HI	09.00	0.00	10.00	1.00	10.00	1.00	10.00	1.00

Design Membership Function Definitions (AFUNCT.DAT)

LO	00.00	1.00	00.00	1.00	00.00	1.00	02.50	0.00
BA	01.00	0.00	03.00	1.00	03.00	1.00	05.00	0.00
AV	03.00	0.00	05.00	1.00	05.00	1.00	07.00	0.00
AA	05.50	0.00	07.50	1.00	07.50	1.00	09.50	0.00
HI	07.50	0.00	10.00	1.00	10.00	1.00	10.00	1.00

Failure Rate Membership Function Definitions (BFUNCT.DAT)

	Ex	Ey	Fx	Fy	Gx	Gy	Hx	Hy
EL	0.00000	0.00	0.01635	1.00	0.01635	1.00	0.03270	0.00
LO	0.01635	0.00	0.02945	1.00	0.02945	1.00	0.04255	0.00
AL	0.02945	0.00	0.04065	1.00	0.04065	1.00	0.05185	0.00
SWL	0.04065	0.00	0.04805	1.00	0.04805	1.00	0.05545	0.00
EBA	0.04805	0.00	0.05680	1.00	0.05680	1.00	0.06555	0.00
BA	0.05680	0.00	0.06260	1.00	0.06260	1.00	0.06840	0.00
SWBA	0.06260	0.00	0.07620	1.00	0.07620	1.00	0.08980	0.00
ABA	0.07620	0.00	0.10105	1.00	0.10105	1.00	0.12590	0.00
AV	0.10105	0.00	0.12990	1.00	0.12990	1.00	0.15875	0.00
AAA	0.12990	0.00	0.16040	1.00	0.16040	1.00	0.19090	0.00
SWAA	0.16040	0.00	0.20530	1.00	0.20530	1.00	0.25020	0.00
AA	0.20530	0.00	0.28745	1.00	0.28745	1.00	0.36960	0.00
EAA	0.28745	0.00	0.51690	1.00	0.51690	1.00	0.74635	0.00
AH	0.51690	0.00	1.00855	1.00	1.00855	1.00	1.50020	0.00
SWH	1.00855	0.00	1.71095	1.00	1.71095	1.00	2.41335	0.00
HI	1.71095	0.00	2.45000	1.00	2.45000	1.00	3.20955	0.00
EH	2.45000	0.00	3.30000	1.00	3.30000	1.00	4.15000	0.00

FAM RULES (RULES.LST)

	LO	SWL	MBA	BA	SWBA	SWAA	AA	MAA	SWH	HI
LO 01	04	03	04	06	07	08	11	12	15	
BA 02	06	06	07	08	09	10	12	13	15	
AV 05	07	08	09	10	12	12	13	14	15	
AA 08	08	11	11	12	13	13	14	15	16	
HI 11	11	12	13	13	14	14	16	17	17	

BIBLIOGRAPHY

1. "Reliability Prediction of Electronic Equipment", MIL-HDBK-217E, Department of Defense: Washington DC, October 1986.
2. "Reliability Modeling and Prediction", MIL-STD-756B, Washington DC: Department of Defense August 1982.
3. James R. Evans, William M. Lindsay, The Management and Control of Quality, St. Paul, Minnesota: West Publishing Company, 1989.
4. Rasool Kenarangui, "Event-Tree Analysis by Fuzzy Probability", IEEE Transactions on Reliability, Vol. 40, No. 1, April 1991.
5. William L. Kubic, Jr., Fred P. Stein, "A Theory of Design Reliability Using Probability and Fuzzy Sets", AIChE Journal, Vol. 34, No. 4, pp. 583-601, April 1988.
6. L. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", IEEE Transactions on Systems, Man, & Cybernetics, Vol. SMC-3, pp. 28-44., January 1973.
7. Kyung S. Park, Ji S. Kim, "Fuzzy Weighted-Checklist with Linguistic Variables", IEEE Transactions on Reliability, Vol. 39, No. 3, August 1990.
8. Bart Kosko, Neural Networks and Fuzzy Systems, Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1992.
9. Zadeh, L. A., "Fuzzy Sets", Information and Control, Vol. 8, pp. 338-353, 1965.
10. Patrick K. Simpson, Artificial Neural Systems, Maxwell House, Fairview Park, Elmsford, New York: Pergamon Press, Inc., 1990.
11. Electronic Equipment Reliability Data (EERD-2), Rome, New York: Department of Defense, Summer 1986.
12. "Editorial", IEEE Transactions on Reliability, Vol. 40, No. 1, April 1991.
13. Naruhito Shiraishi and Hitoshi Furuta, "Reliability Analysis Based on Fuzzy Probability", Journal of Engineering Mechanics, Vol. 109, No. 6, December, 1983.

14. BRAIN MAKER, California Scientific Software: Sierra Madre, CA, January, 1989.
15. Bahrami, Ali, Computer Aided Conceptual Design (CACD): A Hybrid Model For Needs-To-Functions-To-Structures Association By Utilizing The Fuzzy Associative Memory And Fuzzy Information Content, University of Missouri-Rolla: Dissertation, 1992.

VITA

Keith Robert Weyenberg was born on April 28, 1960 in Appleton, Wisconsin. A majority of Mr. Weyenbergs primary education was accomplished in Cameron, Wisconsin. After graduating from Cameron High School, in 1978, he attended the University of Minnesota where he received his Bachelor of Science degree in Aerospace Engineering and Mechanics in December 1982.

In January of 1982 Mr. Weyenberg went to Officer Training School (OTS) at San Antonio, Texas. Upon graduation from OTS in April 1983, he was commissioned a second lieutenant in the United States Air Force and stationed at Hill Air Force Base (HAFB) Utah. At HAFB, Mr. Weyenberg worked as a project, flight test engineer where he was responsible for performance and guidance and control flight tests on several unmanned missile systems. In April 1987, Mr. Weyenberg was promoted to Captain and moved to Las Vegas, Nevada where he assumed the duties of project manager on several classified activities. Mr. Weyenberg was selected by the Air Force to attend graduate school, full time, in October 1990 and enrolled at the University of Missouri-Rolla in August 1991.

Mr. Weyenberg is married to the former Sheila Olson of Cameron, Wisconsin. He also has a daughter, Valerie and son, Michael.